해싱

해싱

해싱이란?

1. 해싱

   - 키 값에 대한 산술적 연산에 의해 테이블의 주소를 계산하여 항목에 접근

2. 해시 테이블

   - 키 값의 연산에 의해 직접 접근이 가능한 자료 구조

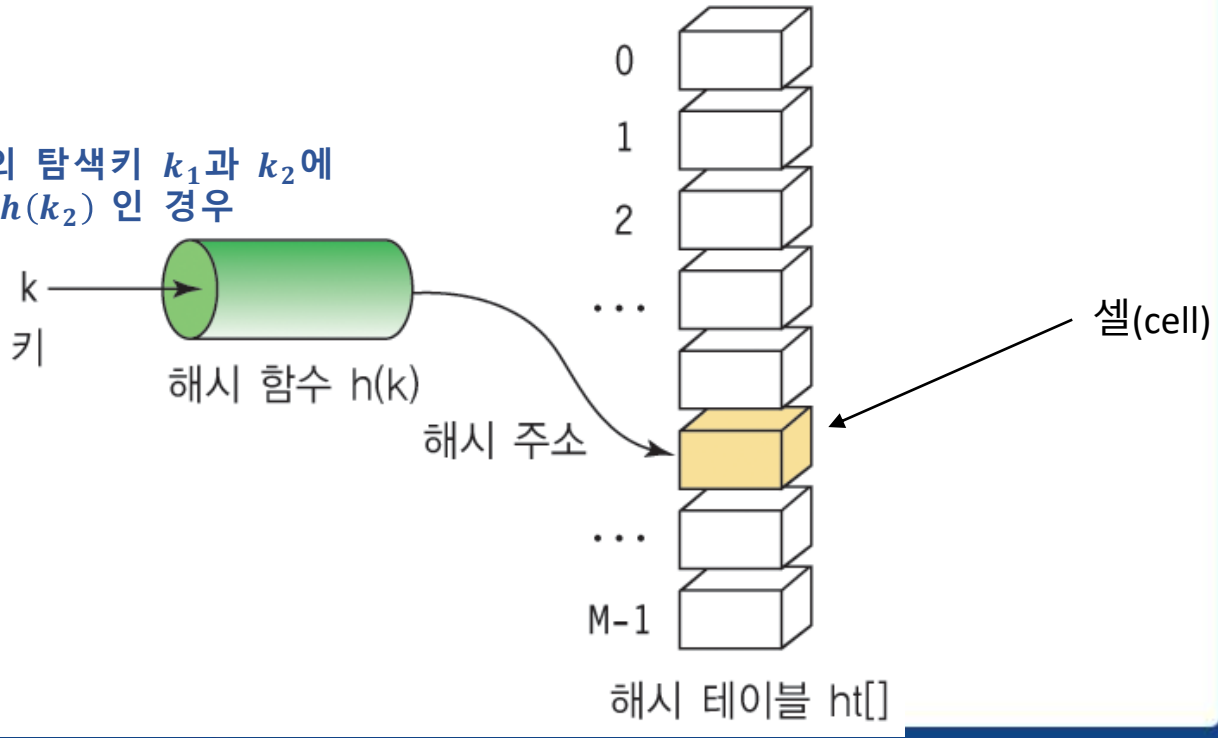
해싱의 원리

해시함수

- 탐색 키(k)를 입력 받아 임의의 정수 $[0, N-1]$에 매핑하는 함수
  - E.g., $h(x) = x \ mod \ N$
- 이를 통해, 해시 주소(hash address) 생성하고 배열로 구현된 해시 테이블의 인덱스로 사용

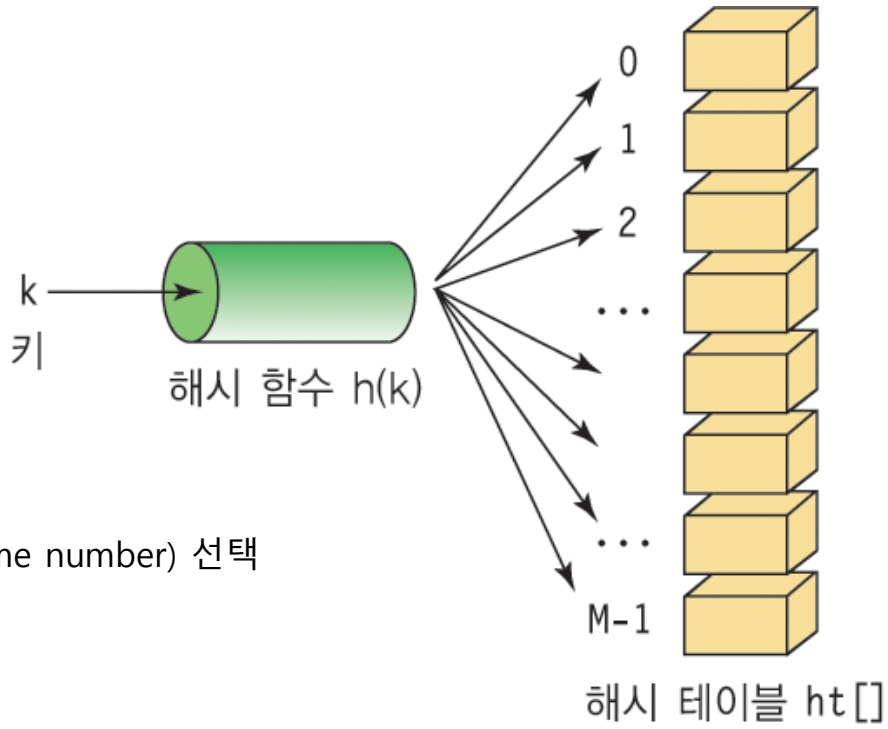
충돌(collision)

- 서로 다른 두 개의 탐색키 k_1과 k_2에 대하여 $h(k_1) = h(k_2)$ 인 경우



해시 함수

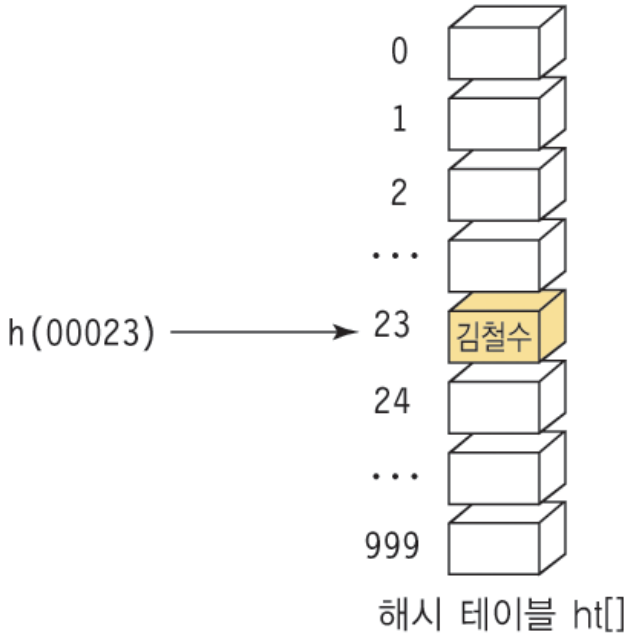
- 좋은 해시 함수의 조건
  - 충돌이 적어야 한다
  - 해시함수 값이 해시테이블의 주소 영역 내에서 고르게 분포되어야 한다
  - 계산이 빨라야 한다



- **제산 함수 (Division function)**
  - h(k)=k mod M
  - 해시 테이블의 크기 M은 **소수**(prime number) 선택

이상적인 해싱

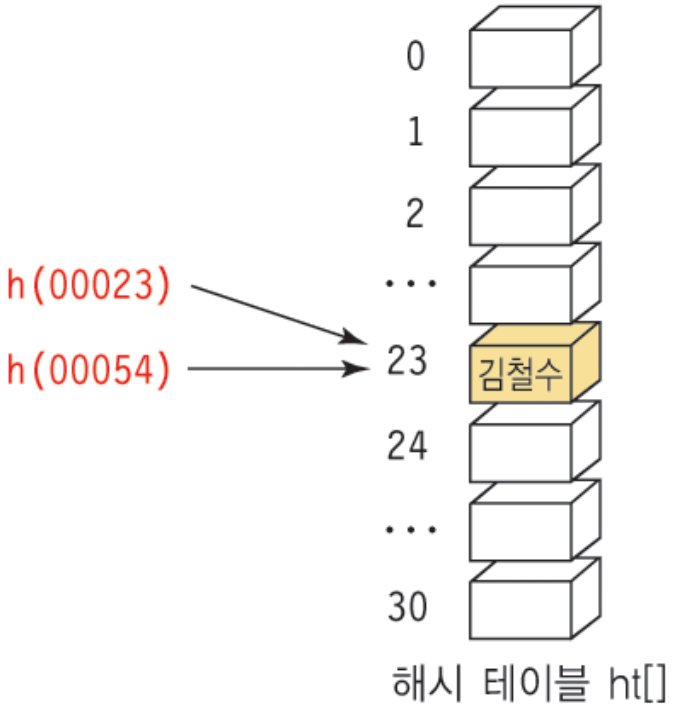
- 학생 정보를 해시 테이블에 저장, 탐색해보자
  - 5자리 학번 중에 앞 2자리가 학과 번호, 뒤 3자리가 각 학과의 학생 번호
  - 같은 학과 학생들만 가정하면 뒤의 3자리만 사용해서 탐색 가능
  - 학번이 00023이라면 이 학생의 인적사항은 셀 ht[23]에 저장
  - 만약 해시테이블이 1000개의 셀을 가지고 있다면 탐색 시간이 **O(1)**이 되므로 이상적임



실제의 해싱

- 실제로는 해시 테이블의 크기가 제한되므로, 존재 가능한 모든 키에 대해 저장 공간을 할당할 수 없음
- $h(k) = k \ mod \ M$ 의 예에서 보듯이 필연적으로 충돌이 발생함

$M = 31$ 일 때,
같은 값으로 해싱이 됨



충돌(Collision)

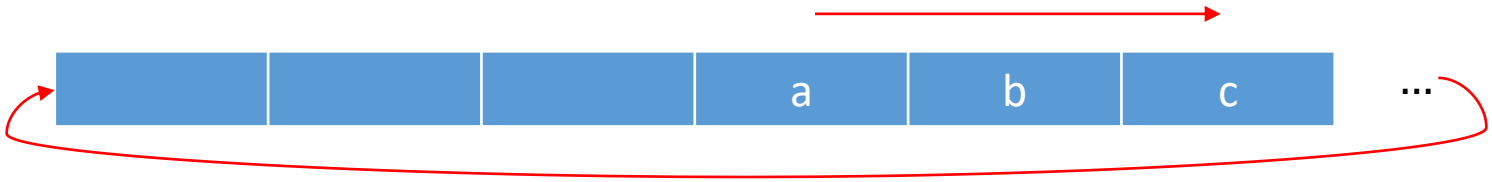
- 서로 다른 탐색 키를 갖는 항목들이 같은 해시 주소를 가지는 현상
- 충돌이 발생하면 해시 테이블에 항목 저장 불가능
- 충돌을 효과적으로 해결하는 방법 반드시 필요

충돌해결책

- 선형조사법, 이중해싱법: 충돌이 일어난 항목을 해시 테이블의 다른 위치에 저장
- 체이닝: 각 셀에 삽입과 삭제가 용이한 연결 리스트 할당

선형조사법(Linear probing)

- 충돌이 $ht[i]$에서 발생할시
  - $i = k \ mod \ M$ (Key 값에 의해 계산되는 인덱스)
  - $ht[i+1]$이 비어 있는지 조사
  - 비어있는 공간이 나올 때까지 계속 조사
  - 테이블의 끝에 도달하게 되면 다시 테이블의 처음부터 조사
  - 조사되는 위치: $ht[i]$ -> $ht[i+1]$ -> $ht[i+2]$ ->...



선형조사법(Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	6

Step 1 : $h(8) = 8 \; mod \; 7 = 1$

Step 2 : $h(1) = 1 \; mod \; 7 = 1$

Step 3 : $h(9) = 9 \; mod \; 7 = 2$

Step 4 : $h(6) = 6 \; mod \; 7 = 6$

Step 5 : $h(13) = 13 \; mod \; 7 = 6$

이중해싱법(Double hashing)

- 충돌이 발생하면 추가적인 해시 함수 사용
- $i = h(k) = k \bmod M,\ \ d(k) = N - (k \bmod N)$
- $i_j = (i + jd(k)) \bmod M$
- 조사되는 위치 : $h[k]$ -> $h[i_1]$ -> $h[i_2]$ -> ...

이중해싱법(Double hashing)

- 예) $h(k) = k \ mod \ 7, d(k) = 5 - (k \ mod \ 5)$ 일 때, (8, 1, 9, 6, 13) 삽입

	값
[0]	
[1]	8
[2]	9
[3]	13
[4]	
[5]	1
[6]	6

Step 1 : $h(8) = 8 \ mod \ 7 = 1$

Step 2 : $h(1) = 1 \ mod \ 7 = 1$ (충돌)
$d(1) = 5 - (1 \ mod \ 5) = 4$

Step 3 : $h(9) = 9 \ mod \ 7 = 2$

Step 4 : $h(6) = 6 \ mod \ 7 = 6$

Step 5 : $h(13) = 13 \ mod \ 7 = 6$ (충돌)
$d(13) = 5 - (13 \ mod \ 5) = 2$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	6

Step 1 : $h(8) = 8 \; mod \; 7 = 1$

Step 2 : $h(1) = 1 \; mod \; 7 = 1$

Step 3 : $h(9) = 9 \; mod \; 7 = 2$

Step 4 : $h(6) = 6 \; mod \; 7 = 6$

Step 5 : $h(13) = 13 \; mod \; 7 = 6$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \, mod \, 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	6

Delete step 1 : $h(6) = 6 \, mod \, 7 = 6$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	

Delete step 1 : $h(6) = 6 \; mod \; 7 = 6$

Delete step 2 : $h(13) = 13 \; mod \; 7 = 6$

탐색 값이 없습니다. (**탐색 실패**)

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	6

Step 1 : $h(8) = 8 \; mod \; 7 = 1$

Step 2 : $h(1) = 1 \; mod \; 7 = 1$

Step 3 : $h(9) = 9 \; mod \; 7 = 2$

Step 4 : $h(6) = 6 \; mod \; 7 = 6$

Step 5 : $h(13) = 13 \; mod \; 7 = 6$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \, mod \, 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	6

Delete step 1 : $h(6) = 6 \, mod \, 7 = 6$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	AVAILABLE

← 다음 값 탐색

Delete step 1 : $h(6) = 6 \; mod \; 7 = 6$

Delete step 2 : $h(13) = 13 \; mod \; 7 = 6$

해시테이블의 삭제 연산 (E.g., Linear probing)

- 예) $h(k) = k \; mod \; 7$ 일 때, (8, 1, 9, 6, 13) 삽입 후, (6, 13) 삭제

	값
[0]	13
[1]	8
[2]	1
[3]	9
[4]	
[5]	
[6]	AVAILABLE

탐색 성공

Delete step 1 : $h(6) = 6 \; mod \; 7 = 6$

Delete step 2 : $h(13) = 13 \; mod \; 7 = 6$

해싱 vs 다른 탐색 방법

탐색 방법		탐색	삽입	삭제
순차 탐색		$O(n)$	$O(1)$	$O(n)$
이진 탐색		$O(\log_2 n)$	$O(\log_2 n + n)$	$O(\log_2 n + n)$
이진 탐색 트리	균형 트리	$O(\log_2 n)$	$O(\log_2 n)$	$O(\log_2 n)$
	경사 트리	$O(n)$	$O(n)$	$O(n)$
해싱	최선의 경우	$O(1)$	$O(1)$	$O(1)$
	최악의 경우	$O(n)$	$O(n)$	$O(n)$

코드

```cpp
#include<iostream>
#include<vector>

#define MAX 353333
#define NOITEM 0
#define ISITEM 1
#define AVAILABLE 2

using namespace std;

class cell {
public:
    int key;
    int value;
    int flag;
    cell() {
        key = -1;
        value = -1;
        flag = NOITEM;
    }
    ~cell() {}
};
cell hashArr[MAX];
int sz = 0;

int hashfunc(int idx) {
    return idx % MAX;
}

int hashfunc2(int idx) {
    return (17 - (idx % 17));
}
```

코드

```
33
34    void tableInsertLinear(int key, int value){ ... }
42
43    void tableSearchLinear(int key, int value){ ... }
55
56    void tableDeleteLinear(int key, int value){ ... }
72
73    void tableInsertDouble(int key1, int key2, int value){ ... }
81
82    void tableSearchDouble(int key1, int key2, int value){ ... }
94
95    void tableClear(){ ... }
102
```


Thank you